Text Sentiment Analysis: Exploring Data Augmentation

Abstract

Deep learning-based studies of text sentiment analysis (TSA) usually desires a labeled large dataset in order to produce significant performance in a domain. However, such demand in training data is often not satisfied given how expensive and tedious it is to produce meaningful labeled text. To overcome the challenge of doing sentiment classification on small dataset, data augmentation techniques are frequently used to obtain new training texts through modifying the existing data. In this assignment, we aim to investigate how data augmentation affects the training of deep learning models. With three rule-based and three deep-learning-based augmentation methods applied on each of the three deep language models chosen (i.e. CNN, BiLSTM, BERT), it is found that different model responses differently on different augmentation methods.

1 Introduction

1.1 Data preparation

The IMDb text classification dataset contains 50,000 text movie reviews with sentiment labeled as positive or negative. The texts are preprocessed following the below steps:

1. convert letters to lower case
2. tokenize sentences
3. remove left and right space of the tokenized words
4. remove stop words given by NLTK
5. remove HTML content
6. remove brackets

After the above cleaning process, the dataset is divided into 35,000 training data and 7,500 development data and 7,500 testing data respectively.

2 Model Design and Implementation

In this section, we will discuss the model architectures designed: Bidirectional Long Short-Term Memory (BiLSTM) layer, convolution layer (CNN) and applied transfer learning using pre-trained BERT Transformer.

2.1 Recurrent Neural Network based BiLSTM

As we know, Recurrent Neural Network (RNN) is designed to process sequential information, that is, data presented in a sentence as it attempts to capture dependency among the data points in the sequence. Nevertheless, RNNs work upon the fact that the result of an information is dependent on its previous state or previous n time steps and have difficulties in learning long range dependencies resulting from the vanishing and exploding gradient problems. While LSTM has the ability to add and remove information to the cell states through gates, it is able to better preserve long term useful information. Bidirection LSTM,or BiLSTM in short, has 2 networks, one has the access to past information in forward direction, and another one has the access to future information in the reverse direction. BiLSTM is a more appropriate model for sentiment analysis in movie reviews as the review data provided by customers usually involves more interlinked contextual information in both directions. Therefore, we designed a RNN-based Bidirectional LSTM model to perform sentiment classification.

2.1.1 Architecture

Data preprocessing is necessary to feed the text data into a deep learning model. The tokenizer was fit on training sentences with a vocabulary size of 6,000. We converted all training sentences, validation sentences and test sentences to tokenized sequences and padded with post padding type with maximum length of 500 for each sequence. In our LSTM model, we added an embedding layer with input vocabulary size of 6,000 after EDA, output length of 16 for each word and maximum length of 500 of input sequence.

Figure 1: BiLSTM Architecture

2.1.2 Hyper-parameter tuning

In order to obtain the best set of hyper-parameters for best performance in test data, hyper-parameter tuning using grid search method was performed on 5 parameters: batch size, learning rate, optimizer, dropout rate and number of dense units.
With each unique set of hyper-parameters, we trained the model on all training data and used validation data to check for the performance of the model. We obtained the parameters with the highest validation accuracy in training process and the best parameter set obtained is as follows: batch size 64, learning rate 0.005, dropout rate 0.5, number of dense units 32 with Adam optimizer.

2.2 Convolution Neural Network

From the previous section, one can tell that RNN-based models, in our case, BiLSTM in particular, is able to achieve higher accuracy in sentiment classification; however, the training time is long as all RNN-based models need to perform the training process sequentially, which makes it not parallelable.
CNN, on the other hand, is famous for its efficiency in training as it permits parallel training. Therefore, CNN in sentiment classification is worth exploring.

2.2.1 Architecture

Since there is no pre-trained tokenizer used in this section, we adopted the same data preprocessing method as the previous section on BiLSTM model. Similarly, in the CNN model, we added the embedding layer with input vocabulary size of 6,000 after EDA, output length of 16 for each word and maximum length of 500 of input sequence. The convolution part consists of a dropout layer,a 1D convolution layer with number of filters, kernel size, stride size and padding to be tuned and a global max pooling layer to obtain the features. It is followed by a dense layer and an output layer.

Figure 2: CNN Architecture

2.2.2 Hyper-parameter tuning

In order to obtain the best set of hyper-parameters for best performance in test data, hyper-parameter tuning using grid search method was performed on the following parameters: number of filters, kernel size, stride size, number of dense units, kernel initializer, dropout rate, optimizer, batch size and learning rate.
With each unique set of hyper-parameters, we trained the model on all training data and used validation data to check for the performance of the model. We obtained the parameters with the highest validation accuracy in training process and the best parameter set obtained is as follows:

filters	kernel	stride	dense units	kernel initializer	drop out rate
256	5	1	512	TruncatedNormal	0.25

Table 1: CNN Architecture Hyper-parameter

optimizer	learning rate	batch size	# epoch
RMSprop	0.001	64	10

Table 2: CNN Training Hyper-parameter

2.3 Transfer Learning by Transformers

Similar to RNN, the transformers are designed to handle sequential input data. However, the attention mechanism that transformers adopts does not require it to process the data in order. In long and complex text data, RNN may lose some essential sentiment information while the transformer can identify the context and avoid having to process the text from start to end.

2.3.1 Architecture

In the experiment, the pretrained model - Bidirectional Encoder Representations from Transformers (BERT) was trained to extract the sentiment of the reviews. At the pretraining stage, BERT was trained for Masked Language Model and Next Sentence Prediction to capture the text representation at both word and sentence level. As such, the model is equipped with the ability to capture bidirectional context information, therefore capturing word and sentence dependency more efficiently.
The BERT Model transformer TFBertForSequenceClassification was loaded and fine-tuned.

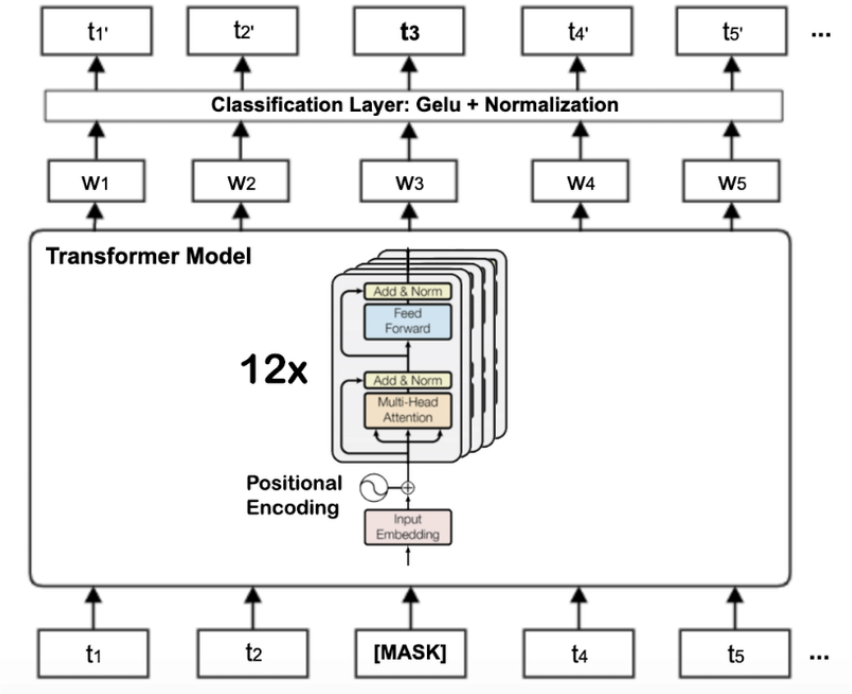

Figure 3: BERT Architecture

2.3.2 Hyper-parameter tuning

In the hyperparameter tuning of the BERT model, we focused on searching for the learning rate, as well as the batch size of the optimal model. It has been found that while a lower learning rate is needed to overcome the catastrophic forgetting problem, an overly small learning rate fails to allow the model to converge. This explains why the optimal learning rate found from fine tuning was 3e-5, which is the medium of the parameter range that the original paper adopted.[1] At the same time, the batch size 32 used in the original paper was found to be an optimal size in our experiment at well.

model	test accuracy	training time (s) / epoch
BiLSTM	0.8860	44
CNN	0.8893	10
BERT	0.8872	582

Table 3: Empirical result on 3 models

2.4 Empirical Analysis

It can be seen that while the three models have similar accuracy performance on the full IMDb movie review dataset, CNN is significantly more efficient than the other two models in terms of its training time per epoch.

3 Data Augmentation

Training deep learning model needs a large amount of data. However, we sometimes may not have enough data in the view of diversity or quantity. To improve the generalization ability or improve the performance of the model, data augmentation is an important technique.
In order to investigate the impact of data augmentation, we propose to subsample the 35,000 training data and apply various techniques of augmentation to increase the number of training samples respectively. We will evaluate the performance of the 3 models on the augmented data individually and draw conclusions.

3.1 Data subsample

We randomly subsample 20% of the training set we discussed before as the new training set, which gives us 7000 instances. The validation and test set remain unchanged. The new training set will be augmented to the size of original training set (35000 instances).

model \ method	small sample	duplicates small sample (x5)
BiLSTM	0.8265	0.8460
CNN	0.8551	0.8431
BERT	0.8653	0.8723

Table 4: Test accuracy for benchmark augmentation methods on 3 different models

From the table, one can easily tell that simple data augmentation through repetition of training samples works badly for CNN, where the test accuracy drops from 0.8551 with small sample to 0.8431 with duplicates of the small sample. This is possibly because for CNN, the model has suffered from overfitting and is unable to generalise in test data. For BiLSTM and BERT, one can observe that the test accuracy increases for model trained with duplicates as compared to that trained with the small sample. This is possibly due to the fact that these models can still extract useful information without overfitting.

3.2 Rule-based methodology

We mainly adopted the methods provided in NLPAUG library to perform rule-based augmentation on our subsampled data. Due to the fact that rule-base augmentation usually does not

need to understand the meaning of the context, it is applied to character level and some of the word-level data generation, where the complete contextual understanding is not necessary.

3.2.1 Character-level Augmentation

We used 3 main methods to perform character-level augmentation on the subsampled data, namely optical character recognition , keyboard augmentation and random character augmentation.

Optical Character Recognition

The OCR augmenter simulates the errors produced when converting text from image based inputs. Even though our input dataset is from IMDb reviews which are not text recognised from images, in order to generalise the data for all possible scenarios, we include this method as one of the character-level augmentation.
Particularly, the method will change some of the characters in the sentence to those that look similar to them. For example, the character 'o' will be changed to the integer '0' after the process.

Keyboard Augmenter

The keyboard augmenter will augment data to target the possible error of keyboard input typo issues. In particular, this method will substitute character by keyboard distance. For example, on keyboard, the character 'm' is close to 'n', so synthetic data will substitute some of the 'm' with 'n'.

Random Augmenter

The random augmenter also simulates the event of keyboard input typo. Some of the techniques include randomly insert characters, substitute characters, swap characters, and delete characters.

Combined Augmenter

We performed a random selection of the character-level augmenters and generated 4 more instances for each of the 7000 subsampled training sentences. Corresponding labels are appended to the 7000 labels. We now have 35000 training samples.

3.2.2 Word-level Augmentation

For word-level augmentation using rule-based method, we adapted the approach of word embedding augmenter with GloVe, TF-IDF augmenter and synonym or antonym augmenter. Note that in this part, we only use non-contextual word-level augmenters. This is because in rule-based methods, the augmenter does not understand the context or the meaning of the full sentence, it only has simple understandings or interpretations of the word itself.

Word embedding augmenter with GloVe

In this method, we use pre-trained embedding GloVe to insert and substitute similar word to the words in the sentences.

TF-IDF augmenter

In this method, we insert and substitute word by TF-IDF similarity.

Combined embedding augmenter and TF-IDF augmenter

We combine both word embedding augmenter with GloVe embeddings and TF-IDF augmenter to do a random application of these methods. We generated 4 more instances for each of the 7000 subsampled training sentences. Corresponding labels are appended to the 7000 labels. We now have 35000 training samples.

Synonym/antonym augmenter

In this method, we use synonyms and antonyms in wordnet to substitute the words in the sentences to provide data augmentation. As usual, we generated 4 more instances for each of the 7000 subsampled training sentences. Corresponding labels are appended to the 7000 labels. We now have 35000 training samples. Note that when doing antonym augmentation, the corresponding labels of newly generated data cases need to change to 0 if the original data case is labeled as 1, and vice versa. This is because when we use antonyms to augment the words, we reverse the meaning of the sentence largely and the sentiment of the sentence in overall should be reversed as well.

3.2.3 Empirical Analysis for rule-based augmentation

model \ method	char-level	non-contextual word-level	synonym/antonym
BiLSTM	0.8679	0.8572	0.8636
CNN	0.8403	0.8440	0.8553
BERT	0.8888	0.8891	0.5011

Table 5: Test accuracy for rule-based augmentation methods on 3 different models

Comparative experiments were performed on four augmentation methods using three deep language models. Overall, a better result can be obtained when using the BERT model, in comparison with BiLSTM and CNN. When the training is done on the character-level augmented dataset, as well as the non-contextual word-level augmented dataset, the accuracy of the BERT model has shown an increase in test accuracy in compared with the benchmark accuracy (0.8653 from Table 4). However, it is also found that the BERT model does not respond well to the synonym/antonym augmented dataset, with a very low test accuracy of 0.5011.
While it is true that CNN has a better performance and efficiency when handling the full balanced dataset, it did not outperform the other models when handling the augmented dataset. There is a general decrease in performance for both CNN and BiLSTM. A possible interpretation is that the rule-based augmented data fails to retain the semantic meaning of the original text, hence resulting a change in context which further hindered the training where model is trained in a word-by-word manner.
The conclusion drawn from the observation is that overall, at the chracter level and non contextual word-level, the BERT model shows some improvement in test accuracy while the other two did not. With a proper combination of rule-based augmentation method chosen, we believe that it is possible to further improve the accuracy for each model.

3.3 Deep-learning-based methodology

Other than the non-contextual rule-based methods, deep learning based approaches could also be used in text data augmentation. Deep learning methods will have a higher level understanding on the contextual meaning of the given sentences and provide a contextual augmentation instead.

3.3.1 Binning & Mini-batch

Ideally, we should pad/truncate each review to the maximum input length of the model after word tokenization and feed one review to the model each time (i.e. size of mini-batch is 1). However, such method is extremely time-consuming considering that we have total 7000 training instances. So, we need to use mini-batch method to get our argumentation results. Facing the various length of reviews, if we randomly select them to form the mini-batch, instances in that batch will be padded or truncated and lose many useful information.
To reduce the above information lose, we first bin all the training sentences based on their lengths shown in the below histogram:

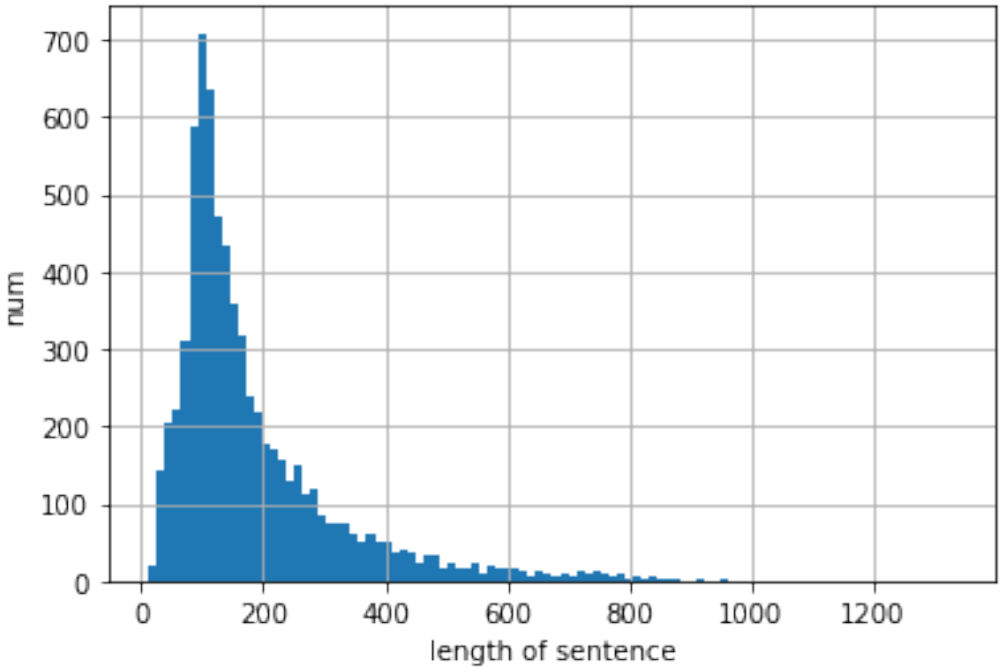

Figure 4: Histogram of Sentence Length of Reviews (bins = 100)

We use all instances in one bin to form one mini-batch, which will lead to minimum information lose. In practical, the number of training instances in one bin may be still quite large for the GPU memory. So, we could further divide each bin into smaller sections, which will be the final mini-batch of our augmentation process.

3.3.2 Word-level Augmentation

In word-level augmentation, we apply substitution of words in the training sentences using RoBERTa language model, which builds on BERT but modifies key hyperparameters, removing the next-sentence pretraining objective and training with much larger mini-batches and learning rates [3].
As the model is trained masked language modeling (MLM) objective, we could mask 20% of the words in our instance and use the model to predict them using the contextual information

of the whole sentence. Then, we select the words with top-5 probability and substitute them to construct our augmentation data.
For example:

I love this <mask> movie and I am very happy.

The model will output *amazing, good, excellent, whole, new* to substitute the word.
Hence, this method, instead of simply finding synonyms or antonyms, the deep learning-based substitution assures a minimum change in semantics while capable of generating unrelated words to broaden the vocabulary of training data.
However, the model computes the top-k words based on semantics instead of sentiment manner, it will reduce the emotional expression the sentence. Also, it could not avoid the problem of ambiguity.

3.3.3 Sentence-level Augmentation

Generalization

The sentence-level generalization augmentation, we use GPT2 [4] language model to generate new texts or summary the texts from the existing ones. In sentence generation, we make the model to generate sentences with length 1000 given by top-k sampling, which is also the GPT2 adopted sampling scheme. For the summary method, we change the output length to 90, which is the half of average length of the sentences in our dataset, and keep the rest the same as before. The examples are obmited here due to the sentence length.
The disadvantage of this method will be the mismatched distribution between the IMDb dataset and the dataset used to train the GPT2 model. So, the given sentences may be not relevant to the movie review and give noises to the original sentences.

Back Translations

Back translation is a method that translates our English data into other language then translates it back to English. We use Marian Machine Translation Model [2] in this task.
We choose Latin, French, Spanish, Italian as inner languages, which is restricted by the fact that Marian Models only trained on Romance languages. Beside the limitation type of inner language (here we only have synthetic languages but not analytic languages like Chinese, Japanese), we will face the limited diversity as we only use one family of translation models.

3.3.4 Empirical Analysis for DL-based augmentation

model \ method	back-translation	sentence-generate	summary	word-substitution
BiLSTM	0.8643	0.8568	0.8389	0.4984
CNN	0.8935	0.8469	0.8537	0.7484
BERT	0.4981	0.4981	0.8777	0.4981

Table 6: Test accuracy for DL-based augmentation methods on 3 different models

Compared to the small sample result we discuss previously, we can see that CNN gets improvement in back-translation dataset and BiLSTM benefits in all augmentation datasets excluding the word-substitution method. We find that BERT cannot convert under the all the above augmentation datasets except the summary one. But it improves slightly in the summary dataset. Also, word-substitution works badly in all three models.

4 Conclusion

The experiments on the original dataset, subsampled dataset, as well as the augmented subsampled datasets, have shown that the proposed augmentation approaches have the capability in altering the training performance both positively and negatively.
We believe that the main reason for the improvements is that the increased diversity on the limited text in the subsampled dataset. Words and sentences are inserted, deleted, substituted, summarised, and back translated randomly or based on the sentence sentiment, so as to obtain a notable number of variations. Such changes have created more instances that are likely to be helpful in model generalization and improving the robustness of it.
However, given the extremely high complexity in natural language, any simple change to the text may alter the semantic meaning and the sentiment of the sentence. An increased noise in the dataset might lead to a lower-quality training.
In our experiment, we have identified that

- for BiLSTM, all implemented augmentation methods are proved effective in enhancing the model performance;
- for CNN, back translation has successfully improved the test accuracy; and
- for BERT, the rule-based augmentation methods are generally more effective than the deep-learning based methods. Out of the methods implemented, summarization, character level and non-contextual word-level augmentation are found to be helpful in improving the test accuracy.

It has proven that we have yet to find an universal applicable augmentation method that enhances the performance of all models.

5 Future Development

To better understand the properties of the augmented data we have (such as distribution between augmented and original data), we may try to understand the latent space (layers before the final output layers) from the trained model by visualizing them on a t-SNE graph.
Moreover, as each model is found to response to different augmentation methods differently, it is possible to attempt a different combinations of the potential methods to find the one that goes the best with the model.

References

[1] Kenton Lee Kristina Toutanova Jacob Devlin, Ming-Wei Chang. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, et al. Marian: Fast neural machine translation in c++. *arXiv preprint arXiv:1804.00344*, 2018.

[3] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[4] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Appendices

- Our experiments are conducted on Colab using **Tesla P100-PCIE-16GB** and NTU GPU Cluster using **Tesla V100-PCIE-32GB**.
- For the hyper-parameter tuning, considering the sufficient time we have, we use grip search. We try 5184 sets of hyperparameters for CNN, 512 for BiLSTM and 6 for BERT.